

Fiches pédagogiques sur l'authentification

CyberEdu



Table des matières

1	Fiche 1 : Mots de passe	5
2	Fiche 2 : Certificats et infrastructures de gestion de clefs	10
3	Fiche 3 : La fédération d'identité	15
4	Fiche 4 : Protocoles d'authentification et d'échange de clé	23
5	Fiche 5 : Authentification de messages	30

Introduction

L'authentification est une procédure permettant pour un système informatique de vérifier l'identité d'une personne ou d'un ordinateur et d'autoriser l'accès de cette entité à des ressources (systèmes, réseaux, applications). L'identification permet de *connaître* l'identité d'une entité, alors que l'authentification permet de *vérifier* l'identité.

Il existe plusieurs *facteurs* d'authentification : utiliser une information que seul le prétendant connaît (mot de passe), possède (carte à puce), est (données biométriques), peut produire (un geste). Les protocoles d'authentification décrivent les interactions entre un *prouveur* et un *vérifieur* et les messages permettant de prouver l'identité d'une entité. On distingue deux familles de protocoles d'authentification : l'authentification simple (un seul facteur d'authentification en jeu) et l'authentification forte (deux facteurs ou plus). Par ailleurs, on parle d'authentification unique lorsqu'un utilisateur n'a besoin de procéder qu'à une seule authentification pour accéder à plusieurs applications informatiques.

Nous avons choisi dans ces fiches de présenter diverses méthodes d'authentification utilisant des mots de passe (fiche 1), des certificats et leur environnement d'infrastructure de gestion de clé (fiche 2) et les mécanismes des fédérations d'identités (fiche 3). Les deux dernières fiches présentent les protocoles d'authentification (fiche 4) et les méthodes d'authentification de données (fiche 5).

Prérequis pour les étudiants

Il n'y a pas de prérequis particulier pour les fiches 1 et 3. Pour les autres, des connaissances de base en cryptographie peuvent être utiles.

Prérequis pour les formateurs

Les fiches apportent des repères pédagogiques aux enseignants, en présentant de manière structurée et concise les sujets importants de certaines notions relatives à l'authentification. Ces fiches ne constituent pas un cours complet sur l'authentification. Il n'est pas demandé à l'enseignant de parfaitement maîtriser le domaine de la sécurité, mais il devra se renseigner sur les sujets présentés pour pleinement exploiter les fiches pédagogiques.

Certaines fiches peuvent demander de présenter quelques éléments de cryptographie de base (en particulier les fiches 4 et 5). Cependant, les exemples proposés peuvent suffire et font partie des bases de la cryptographie.

Utilisation du guide pédagogique

Le but de ce guide est de fournir des fiches pédagogiques pour des enseignants voulant intégrer des notions de sécurité dans des cours sur l'administration des réseaux ou des cours de développement nécessitant des mécanismes d'authentification de personnes ou de documents.

Des liens vers des documents en français ou en anglais sont fournis. Les fiches incluent des références vers des livres, articles et sites web permettant d'approfondir les sujets abordés.

Ci-dessous est proposé un récapitulatif des sujets abordés, ainsi que le temps recommandé pour présenter les sujets et les prérequis correspondants.

Numéro	Sujet	Durée	Prérequis
Fiche 1	Mot de passe	60 minutes	Aucun
Fiche 2	Certificats et IGC	50 minutes	Cryptographie à clé publique : chiffrement et signature
Fiche 3	La fédération d'entité	60-90 minutes	Identifiant numérique (avec attributs)
Fiche 4	Protocoles d'authentification et d'échange de clé	90-120 minutes	Cryptographie à clé publique et à clé symétrique
Fiche 5	Authentification de messages	90-120 minutes	Cryptographie à clé symétrique

1 Fiche 1 : Mots de passe

1.1 Thématique

Thématique	Mots de passe	Numéro de fiche	1	Mise à jour	07/03/2016
------------	---------------	-----------------	---	-------------	------------

1.2 Thème des cours visés

Cette unité d'enseignement présente le concept de mot de passe et son rôle central dans l'authentification.

1.3 Volume horaire

Un volume horaire d'une heure est suffisant pour couvrir le matériel du cours, à l'exception de la partie intitulée « pour aller plus loin ».

1.4 Prérequis / corequis

La connaissance des propriétés attendues d'une fonction de hachage est importante pour comprendre les méthodes de stockage sécurisé de mots de passe.

1.5 Objectifs pédagogiques

Les objectifs pédagogiques principaux du cours sont les suivants :

- introduire le concept de mot de passe et expliquer son rôle central dans l'authentification ;
- donner l'intuition sur l'évaluation de la robustesse d'un mot de passe et présenter le compromis sécurité-facilité d'utilisation au niveau du choix du mot de passe ;
- discuter le problème du stockage et de la protection des mots de passe.

1.6 Conseils pratiques

Afin que le concept de mot de passe ne reste pas abstrait pour l'étudiant, l'enseignant est encouragé à donner des exemples concrets de mot de passe, par exemple celui d'un mot de passe sur 8 caractères détaillé un peu plus loin. Les conseils de bonne d'hygiène informatique abordés un peu plus loin dans la fiche devraient aussi être rappelés aux étudiants. De même, l'enseignant devrait aussi insister sur certains conseils qui relèvent du bon sens, comme le fait de ne jamais partager son mot de passe avec un camarade, ou encore de ne pas le donner à un outil en ligne pour vérifier sa robustesse.

Cette fiche présente peu de valeurs chiffrées car elles dépendent essentiellement de la configuration utilisée par l'attaquant (CPU, CGU etc.), du contexte (local ou en ligne) et des algorithmes que l'on étudie (MD5, SHA-1, LM, etc.). Pour éviter les écueils liés à des chiffres qui pourraient être approximatifs, l'enseignant pourra utiliser des outils de *benchmark* pour faire prendre conscience des ordres de grandeurs liés aux configurations des étudiants (par exemple, utiliser la commande test de *John The Ripper*).

1.7 Description

Authentification par mot de passe. Le mot de passe est au cœur de la sécurité de nombreux systèmes et parfois le seul mécanisme de protection. L'objectif principal de l'authentification par mot de passe est de pouvoir vérifier l'identité d'un utilisateur avant de lui donner accès à des informations (p. ex. son courrier électronique ou ses documents) ou des ressources (p. ex. la possibilité d'imprimer sur un serveur distant). Il s'agit d'une preuve d'identité par « ce que l'on sait » par contraste avec des méthodes d'authentification se basant sur « ce que l'on est » (p. ex. empreinte digitale, iris ou voix) ou « ce que l'on possède » (p. ex. une carte à puce ou un jeton cryptographique). Évidemment si le contexte le justifie, il est possible de combiner l'authentification par mot de passe avec un autre facteur d'authentification pour augmenter le niveau de sécurité globale. Dans ce cas-là, l'authentification sera considérée comme réussie si et seulement si chacune des méthodes d'authentification individuelles est validée.

En pratique, on commence par déclarer son identité (le « *login* » ou l'identifiant) et ensuite l'utilisateur doit renseigner le mot de passe correspondant. Pour qu'il soit sûr, le mot de passe doit être suffisamment imprévisible pour qu'il ne puisse pas être deviné par l'adversaire en utilisant un générateur de dictionnaire ciblé sur la personne par exemple. Cependant, il doit également pouvoir être facilement mémorisé par un être humain ou alors la tentation sera forte de l'écrire sur un support externe, comme un post-it.

Les points importants pour évaluer la sécurité d'une authentification par mot de passe peuvent se résumer par les questions suivantes :

- Comment les mots de passe sont-ils choisis ?
- Comment sont-ils transmis entre l'utilisateur et le vérificateur ?
- Comment sont-ils stockés/protégés par l'utilisateur ?
- Comment sont-ils stockés/protégés par le vérificateur ?

Robustesse d'un mot de passe. Pour qu'un mot de passe soit considéré comme robuste, il faut qu'il soit suffisamment long et aléatoire pour ne pas pouvoir être facilement deviné par l'adversaire. Par exemple, si le mot de passe choisi par un utilisateur est pioché au hasard parmi tous les noms propres et communs existants dans la langue française, il ne faudra que quelques centaines de milliers d'essais au maximum à l'adversaire pour réussir à deviner celui-ci en conduisant une attaque par dictionnaire¹. Cette taille de l'espace des mots de passe n'est clairement pas suffisante pour assurer un bon niveau de sécurité, car un ordinateur est capable de tester plusieurs milliers, voire millions, de mots de passe par seconde selon l'algorithme (MD5, SHA-1, LM...) et les ressources utilisées (CPU, GPU, réseau...). D'un autre côté, si le mot de passe est trop long ou aléatoire, il ne pourra pas être facilement mémorisé et sera donc inutilisable en pratique. En particulier, selon certaines études il est difficile pour être humain de retenir plus de 12 caractères choisis aléatoirement.

Ainsi, si le mot de passe choisi est trop long pour être mémorisé, le risque pratique est que l'utilisateur soit tenté d'écrire le mot de passe en clair sur un autre support, diminuant ainsi la sécurité effective de ce mot de passe. Une alternative raisonnable est de remplacer le mot de passe par une phrase de passe se composant de la concaténation de plusieurs mots faciles à retenir, mais dont l'enchaînement ne doit pas être prévisible.

Afin de quantifier le niveau de sécurité d'un mot de passe particulier, il est possible d'utiliser des

1. Le dictionnaire français compte environ 100 000 mots et le dictionnaire anglais environ 400 000. Pour construire son dictionnaire, un attaquant pourra utiliser ces derniers, ainsi que des listes de mots de passe trouvés sur Internet, qui peuvent rassembler jusqu'à 64 millions d'entrées sur CrackStation

outils de cassage de mots de passe qui cherchent à mesurer à quel point ce mot de passe est prévisible. Par exemple, *John the Ripper* [1] est un logiciel libre de cassage de mots de passe qui peut être utilisé pour évaluer la sécurité d'un mot de passe (p. ex. en cas d'audit). Cet outil permet de tester différentes méthodes de cassage de mots de passe telles qu'essayer des variations sur le *login* de l'utilisateur, mener une attaque par dictionnaire ou encore essayer toutes les combinaisons possibles de caractères. La complexité de l'approche utilisée est bien sûr proportionnelle à la taille de l'espace de recherche exploré par John the Ripper. Par exemple, pour un mot de passe de 8 caractères (sachant qu'il y a en général 95 caractères différents pouvant être écrits grâce à un clavier), la taille de l'espace des mots de passe possible est de l'ordre de $95^8 \approx 6.6 \times 10^{15}$ possibilités². À titre d'exemple, pour cette taille de l'espace des mots de passe, en faisant l'hypothèse que l'adversaire est capable d'essayer 100 000 mots de passe par seconde³, il lui faudra en moyenne environ 104 années pour trouver le bon mot de passe.

Les cas précédents évoquaient la possibilité pour un attaquant de faire une recherche sur son poste local. On parle alors d'attaques hors ligne. Cependant, un attaquant est parfois forcé d'effectuer ses tests directement en ligne. Dans ce cas, il devient possible de limiter son nombre de tentatives en bloquant par exemple le système pour une certaine durée après un certain nombre d'essais. Une autre contre-mesure possible est de s'assurer que chaque essai est bien mené par un humain (et non pas un ordinateur) en utilisant des techniques de type CAPTCHA (dont l'acronyme signifie en anglais « *Completely Automated Public Turing test to tell Computers and Humans Apart* »). Le but d'un CAPTCHA est que le problème sous-jacent doit pouvoir être résolu facilement par un être humain, mais plus difficilement par un ordinateur. Il permet donc de distinguer entre un essai de mot de passe qui est fait par un humain et une tentative de la part d'un script automatisé.

Protection des mots de passe. Dans un système où le mot de passe est le seul moyen d'authentification utilisé, il est très important de s'assurer qu'il est bien protégé de l'adversaire. En particulier, il faut absolument éviter que le mot de passe circule en clair sur le canal de communication entre l'utilisateur et le vérificateur durant le processus d'authentification. Si ce n'est pas le cas, il devient trivial pour un attaquant qui a la capacité d'écouter le canal de communication de pouvoir le rejouer par la suite pour usurper l'identité de l'utilisateur légitime.

En outre, il faut aussi prendre en compte les risques liés au stockage des mots de passe. En effet, l'adversaire n'est pas seulement un attaquant externe qui est capable d'espionner le canal de communication, mais il peut également s'agir de quelqu'un à l'intérieur du système (tel qu'un administrateur système trop curieux); il faut donc absolument éviter que la sécurité du système d'information s'écroule totalement si le serveur stockant les mots de passe n'est pas suffisamment bien protégé. Pour cela, il faut éviter de stocker directement les mots de passe « bruts ». Une solution possible est de stocker pour chaque utilisateur u la paire $(login_u, f(password_u))$ où f est une fonction à sens unique facile à évaluer mais difficile à inverser. Ainsi le vérificateur peut facilement tester la validité d'un mot passe qui lui est présenté, mais le vol du fichier contenant les mots de passe par l'attaquant ne lui permet pas de les retrouver directement. Concrètement, f est généralement construite à partir d'une fonction de hachage cryptographique.

En pratique, si l'adversaire arrive à mettre la main sur un fichier $(login_u, f(password_u))$, il peut essayer de casser certains de ces mots de passe, par exemple en faisant une attaque par dictionnaire. De plus, plutôt que de faire cette attaque par dictionnaire seulement lorsqu'il obtient le fichier de mots de passe, l'adversaire peut à l'avance précalculer la valeur $f(password)$ pour tous les mots

2. Pour avoir un ordre de grandeur, il y a 10^{15} bactéries dans le corps humain.

3. Pour des algorithmes lents et avec un ordinateur peu puissant. Cela peut monter jusqu'à l'échelle du million de mots de passe par seconde pour des algorithmes comme le MD5.

existants dans son dictionnaire. Au moment du cassage, ce précalcul lui permettra de retrouver beaucoup plus rapidement les mots de passe des utilisateurs qui sont contenus dans le dictionnaire que le temps qu'il aurait pris pour faire cette attaque « en ligne ».

Il existe aussi des structures de données telles que les tables arc-en-ciel (*rainbow tables* en anglais) [2] qui permettent d'optimiser le compromis temps/mémoire entre l'espace nécessaire pour stocker les précalculs effectués et le temps requis au moment de l'attaque pour casser les mots de passe (voir aussi [3] pour des structures plus complexes).

Pour compliquer la tâche de l'attaquant, on peut utiliser la technique de salage, qui ajoute une chaîne de bits aléatoires, appelée sel, au mot de passe (qu'il soit en préfixe, suffixe voire au milieu) avant d'appliquer la fonction à sens unique. Si on choisit un sel différent pour chaque utilisateur, cela complique de manière significative la tâche de l'adversaire qui voudrait casser le mot de passe, car il devient impossible de faire les précalculs nécessaires. Un autre avantage du salage aléatoire par mot de passe est que deux mots de passe identiques donneront deux hachés différents.

Une autre possibilité pour ralentir l'adversaire lors du cassage est de rendre coûteuse l'évaluation du haché, p. ex. en utilisant une fonction d'étirement de clé (*key stretching* en anglais) qui par conception est très lente à évaluer. Une telle fonction peut être construite en ajoutant un sel, puis en appliquant une fonction de hachage de manière récursive un très grand nombre de fois (par exemple au moins 10 000 fois sous iOS4 si on utilise PBKDF2 [4]). Afin de donner un ordre de grandeur, en 2011 on estimait qu'un ordinateur standard pouvait faire environ 65 000 évaluations par minute d'une fonction de hachage telle que SHA-1, ce qui consistait à pouvoir dériver 6,5 clés par minute avec PBKDF2 pour le cas d'iOS4. Toutefois ce ralentissement d'évaluation se trouve contrebalancé par l'évolution des performances des machines. Si on prend en compte la loi de Moore (la puissance des ordinateurs double en moyenne tous les un an et demi), pour garder un niveau de sécurité stable il faudrait aussi doubler le nombre d'itérations effectuées par la fonction d'étirement après cette période (ou alors être prêt à accepter la baisse du niveau de sécurité). Ainsi avec les avancées technologiques en 2014, l'adversaire était capable de dériver 24 clés par minute alors qu'en 2017 il sera capable de dériver 104 clés par minute et en 2020 il pourra dériver 416 clés par minute.

Pour résumer parmi les règles de bonne hygiène informatique pour la gestion de mots de passe, on trouve les points suivants :

- changer régulièrement de mots de passe ;
- éviter d'utiliser le même mot de passe dans plusieurs systèmes d'information ou en tout cas avoir un mot de passe différent pour des comptes dont le niveau de sécurité requis est important ;
- utiliser un système qui vérifie un mot de passe candidat et qui détecte s'il est potentiellement facile à deviner ;
- vérifier que le nouveau mot de passe est suffisamment différent de l'ancien ;
- utiliser un système de stockage sécurisé des mots de passe.

Pour aller plus loin. Un des défauts d'un mot de passe « classique » est qu'une fois que l'adversaire a pu mettre la main sur celui-ci, il lui est possible de le « rejouer » autant de fois que souhaité pour s'authentifier. Un mot de passe à usage unique essaye de remédier à ce défaut en liant la validité du mot de passe à une session ou une transaction particulière (p. ex. une transaction bancaire). Ainsi, même si l'adversaire arrive à capturer un mot de passe à usage unique, il lui sera impossible de le réutiliser, car il ne sera plus considéré comme valide.

Un mot de passe unique consiste en une chaîne aléatoire de caractères dont la longueur est souvent plus courte qu'un mot de passe standard. Une façon très simple d'implémenter ce concept est sous la forme d'un tableau papier où chaque case contient une série aléatoire de caractères. Ce

tableau est généré et distribué à l'avance à l'utilisateur et lorsque celui-ci cherche à s'authentifier il doit transmettre la valeur se trouvant au croisement d'une ligne et d'une colonne particulière. Par la suite, le vérificateur ne lui demandera plus jamais la position correspondante.

Une autre manière couramment utilisée est de transmettre le mot de passe à usage unique par un canal auxiliaire qu'on suppose hors de contrôle de l'adversaire. Par exemple, le mot de passe à usage unique pourrait être envoyé par SMS ou par courriel et l'utilisateur devrait être capable de le renvoyer dans un temps prédéfini. Un bénéfice supplémentaire de cette approche est qu'on augmente le niveau de sécurité du système, car on peut aussi vérifier indirectement que l'utilisateur possède le téléphone correspondant (dans le cas du SMS) ou encore qu'il peut accéder au compte de messagerie électronique.

Enfin, d'autres implémentations sont possibles telles que l'utilisation d'un jeton cryptographique. Ce jeton possède un secret (telle que la graine d'un générateur pseudo-aléatoire) qui est connu aussi du vérificateur et est synchronisé avec lui. Chaque mot de passe qui est obtenu grâce au générateur pseudo-aléatoire du jeton n'est valable que pendant une courte période et doit être ensuite envoyé par l'utilisateur.

1.8 Matériels didactiques et références bibliographiques

- [1] JOHN THE RIPPER PASSWORD CRACKER, <http://www.openwall.com/john/>
- [2] PHILIPPE OECHSLIN, *Making a Faster Cryptanalytic Time-Memory Trade-Off*, CRYPTO 2003, pp. 617-630. http://link.springer.com/chapter/10.1007%2F978-3-540-45146-4_36
- [3] GILDAS AVOINE, ADRIEN BOURGEOIS, XAVIER CARPENT, *Analysis of Rainbow Tables with Fingerprints*, ACISP 2015, pp. 356-374. http://link.springer.com/chapter/10.1007%2F978-3-319-19962-7_21
- [4] BURT KALISKI, *PKCS 5 : Password-Based Cryptography Specification Version 2.0*. <http://tools.ietf.org/html/rfc2898#section-5.2>

2 Fiche 2 : Certificats et infrastructures de gestion de clefs

2.1 Thématique

Thématique	Certificats	Numéro de fiche	2	Mise à jour	07/03/2016
------------	-------------	-----------------	---	-------------	------------

2.2 Thème des cours visés

Cette unité d'enseignement présente les certificats de clefs publiques et les infrastructures de gestion de ces clefs. Elle peut être par exemple intégrée dans un cours d'administration système, de réseaux informatiques, ou de développement web.

2.3 Volume horaire

L'enseignement comporte quatre sujets qui seront présentés dans l'ordre suivant :

- fondamentaux sur les certificats : 15 minutes ;
- infrastructure de gestion de clefs : 15 minutes ;
- certificats X.509 : 10 minutes ;
- clefs PGP : 10 minutes.

La partie sur les fondamentaux aborde les certificats avec une approche académique. Elle est essentielle à la compréhension du sujet. Il est également fortement recommandé de présenter la partie sur les infrastructures de gestion de clefs, qui utilise également une approche académique tout en introduisant des contraintes réelles (notamment la nécessité d'utiliser des autorités d'enregistrement en sus des autorités de certification). La troisième partie sur les certificats X.509 permet de relier le cours aux préoccupations réelles des étudiants. Enfin, la quatrième partie sur PGP n'est pas essentielle pour la compréhension des certificats. Elle apporte toutefois un éclairage intéressant si PGP est présenté aux étudiants dans le cadre d'un autre chapitre.

2.4 Prérequis / corequis

La connaissance de la cryptographie à clef publique est un prérequis à cette unité d'enseignement. Plus précisément, les étudiants doivent connaître le concept de couple clef publique/privée. Ils doivent également savoir comment utiliser ces clefs soit dans le cas du chiffrement, soit dans le cas de la signature.

2.5 Objectifs pédagogiques

L'objectif pédagogique est de faire comprendre aux étudiants les fondements d'un certificat. Ils doivent notamment assimiler le fait qu'un certificat n'a de sens que parce qu'il existe une relation de confiance – typiquement par l'intermédiaire d'un tiers de confiance – entre les parties. À l'issue de l'enseignement, les étudiants doivent notamment comprendre ce que signifie « vérifier le certificat d'un site web ».

2.6 Conseils pratiques

Il est usuel de côtoyer des étudiants de dernière année d'informatique qui ne savent pas ce qu'est un certificat. Ils utilisent tous les jours des sites web protégés avec HTTPS sans pour autant s'interroger sur les principes fondamentaux de ce protocole. Dans le meilleur des cas, les étudiants répondent qu'il faut « cliquer sur le cadenas » pour savoir si le site est authentique. Durant cet enseignement, il faut apprendre aux étudiants trois fondamentaux, à savoir :

- le contenu d'un certificat (clef publique, identité du porteur, signature de l'autorité),
- la manière de vérifier un certificat (signature de l'autorité, non-révocation, date de validité éventuelle),
- l'importance du fait que c'est l'utilisateur qui doit générer lui-même sa clef privée (sauf cas des autorités de séquestre).

2.7 Description

Fondamentaux

L'enseignant doit avant tout expliquer aux étudiants que (1) le rôle d'un certificat est de lier une *clef publique* à l'*identité de son propriétaire*, et que (2) ce lien est assuré par une tierce partie de confiance par le biais d'une signature numérique portant sur ces deux données. Il est fondamental que l'enseignant s'assure de la bonne compréhension de ce concept avant de passer à la suite.

L'enseignant pourra introduire la terminologie et ainsi préciser que la tierce partie de confiance est appelée *autorité de certification*. Elle certifie que la clef publique appartient bien au dit propriétaire en délivrant un *certificat* qui contient avant tout :

- la clef publique ;
- l'identité du propriétaire ;
- la signature de l'autorité de certification sur ces deux. données.

Pour vérifier le certificat d'une personne, il faut vérifier que la signature de l'autorité calculée à la fois sur la clef publique et sur l'identité de son propriétaire déclaré (ainsi qu'éventuellement sur des informations complémentaires) est correcte. Cette vérification nécessite de faire confiance à l'autorité et de connaître sa clef publique.

L'enseignant devra introduire la hiérarchie des clefs et parler de la racine de la confiance en décrivant la cascade de vérification de certificats. L'enseignant insistera sur l'importance pour un utilisateur de s'assurer de l'authenticité de la clef publique racine qu'il possède : « cette clef que je vais utiliser pour vérifier des certificats est-elle bien la clef publique de l'autorité à qui je fais confiance ? » L'enseignant pourra également introduire le concept de certificat racine (*root certificate*) : il s'agit du certificat qui contient la clef publique racine et qui est signé par la clef privée correspondant à cette clef publique. On parle alors de certificat auto-signé.

L'enseignant pourra prendre l'exemple de HTTPS : très peu d'étudiants savent que le navigateur contient un grand nombre de certificats racines. Il est alors essentiel de rappeler l'importance d'installer dans son système un navigateur légitime (possibilité de parler de la vérification de l'empreinte du navigateur).

Il est également important de bien faire comprendre aux étudiants que la possession d'un certificat ne peut pas à elle seule procurer de la sécurité : c'est ce qui est fait de la clef privée, associée à la clef publique certifiée qui permettra d'assurer, par exemple, l'authenticité d'une personne ou d'un

document. Certains étudiants pensent en effet qu'il suffit de présenter son certificat – comme on le ferait avec une carte d'identité – pour prouver son identité.

Infrastructure de gestion de clefs

L'enseignant présentera dans cette section les différents éléments constituant une infrastructure de gestion de clefs (*Public Key Infrastructure*, PKI) : autorité de certification, autorité d'enregistrement, autorité de dépôt et autorité de séquestre.

Autorité de certification : entité qui génère les certificats, c'est-à-dire qui signe les données, à savoir la clef publique et l'identité de son propriétaire, ainsi que d'éventuelles informations complémentaires, comme le numéro du certificat et sa date de validité. L'enseignant pourra donner des exemples des principales autorités de certification existantes.

Autorité d'enregistrement : entité qui gère l'authentification des personnes requérant un certificat. L'enseignant expliquera que les autorités de certification délèguent l'authentification du demandeur à une tierce partie appelée autorité d'enregistrement. L'enseignant pourra alors introduire le concept de *certificate signing request* qui est le document transmis par l'autorité d'enregistrement au demandeur, et que ce dernier doit transmettre à l'autorité de certification. L'enseignant pourra également discuter des différents niveaux de certification en fonction du moyen d'authentification utilisé : authentification reposant sur un simple échange de messages électroniques, sur une photocopie de la pièce d'identité, etc.

Autorité de dépôt : entité qui stocke et met éventuellement à disposition les certificats générés par l'autorité de certification. Plus fondamental, l'autorité de dépôt diffuse également la liste de révocation des certificats (CRL).

Autorité de séquestre : optionnelle, cette entité peut conserver en séquestre une copie de la clef privée associée au certificat délivré. Ce type de séquestre peut être utile pour des clefs privées de chiffrement dans le cadre d'une entreprise. Cela permet à l'employeur d'avoir accès aux courriers professionnels de d'un employé après son départ de l'entreprise.

L'enseignant précisera que chacun peut mettre en place sa propre infrastructure de gestion de clefs dans un cercle fermé, c'est-à-dire dans un cercle qui lui accorde sa confiance, par exemple dans une entreprise ou au sein d'un groupe d'amis.

Certificats X.509

Après que les étudiants ont acquis les connaissances fondamentales sur les certificats, l'enseignant pourra passer à la pratique en décrivant les éléments essentiels de la norme X.509. Pour cela, l'enseignant pourra utiliser un véritable certificat X.509, par exemple le certificat du « *webmail* » de l'établissement où a lieu l'enseignement ou le certificat d'un passeport électronique pour montrer qu'il n'y a pas que HTTPS qui utilise des certificats. L'enseignant mettra en évidence les principaux champs et insistera notamment sur la date de validité et l'importance de cette dernière dans le cas où la clef privée associée au certificat est perdue. Il indiquera également qu'un certificat est émis pour un usage donné, par exemple pour vérifier des signatures mais pas pour chiffrer des messages.

Lorsque tous ces concepts auront été assimilés, l'enseignant pourra synthétiser la vérification d'un certificat (qui peut être en tout ou partie réalisée automatiquement par le navigateur, dans le cas de HTTPS par exemple) :

- vérification de la signature de l'autorité de certification,
- vérification de la date de validité,
- vérification que le certificat est utilisé pour l'usage pour lequel il a été certifié,
- vérification que le certificat n'est pas révoqué.

Vérifier qu'un certificat n'est pas révoqué est trop souvent négligé. Il s'agit pourtant d'une vérification fondamentale car le possesseur d'une clef publique doit pouvoir la révoquer si la clef privée correspondante a été corrompue ou perdue, ou si l'utilisateur a perdu les droits d'utiliser la clef privée pour l'usage prévu (par exemple, un employé ne doit plus être en mesure de signer des documents au nom de son entreprise après l'avoir quittée). Une infrastructure de gestion de clefs doit par conséquent mettre à disposition une liste de révocation des certificats (CRL, *Certificate Revocation List* en anglais). Une telle liste est un document contenant les identifiants des certificats émis par l'autorité qui ont été révoqués par leur propriétaire. La raison de la révocation est parfois adjointe à l'identifiant du certificat.

Dans le cas d'une chaîne de certificats, un exercice intéressant est de faire calculer aux étudiants quelle doit être la durée de validité du certificat racine. C'est un exercice utile car on s'aperçoit par exemple dans le cas du passeport électronique qu'il y a des pays qui commettent des erreurs lors du calcul de cette date et qui mettent des dates de validité trop courtes pour les certificats racines. Il est possible d'illustrer ce problème sous la forme d'un exercice : « La puce d'un passeport électronique contient des données ainsi qu'une signature numérique calculée sur ces données. Le ministère de l'Intérieur utilise la même clef privée pour signer tous les passeports émis dans un intervalle de 3 mois : une nouvelle clef est en effet générée tous les trois mois pour signer les futurs passeports. Les clefs publiques correspondantes sont diffusées dans des certificats signés avec la clef privée racine du ministère de l'Intérieur. Cette clef privée racine est remplacée tous les 5 ans. Quelle doit être la durée de validité du certificat racine pour que les citoyens puissent utiliser sans soucis leur passeport pendant 10 ans ? » Il est fondamental qu'un passeport puisse être vérifié pendant toute sa durée de validité, qui est de 10 ans. Sachant que la clef privée du ministère de l'Intérieur est utilisée pendant 3 mois, le certificat correspondant doit être valide pendant au moins 10 ans et 3 mois. Pour vérifier ce certificat, il faut que le certificat racine soit valide. La clef privée correspondant à ce dernier étant utilisée pendant 5 ans, il faut que le certificat racine soit valide pendant 5 ans ajoutés au 10 ans et 3 mois, soit 15 ans et 3 mois.

Clefs PGP

Si l'outil PGP a été présenté aux étudiants, alors l'enseignant pourra comparer l'approche X.509 avec l'approche PGP en termes de gestion de la confiance. Dans le premier cas, il s'agit de structures hiérarchiques où chacune d'entre elles se termine avec un enracinement unique de la confiance, alors qu'il s'agit d'une confiance distribuée dans le second cas.

La confiance avec PGP repose en effet sur des relations pair à pair (non nécessairement symétriques) entre utilisateurs : un utilisateur A fait confiance à un utilisateur B s'il considère que B applique les règles de l'art pour certifier les clefs publiques d'autres utilisateurs du système. Alors que l'on parle de *clefs publiques PGP*, il faut souligner qu'il s'agit en fait de *certificats* signés par les pairs. Une clef publique peut ainsi être certifiée par plusieurs pairs, contrairement à ce qui est pratiqué dans une structure hiérarchique.

L'enseignant pourra enfin présenter le graphe de confiance de PGP et discuter de ses limites. L'un des principes de PGP est en effet la transitivité de la confiance : si un utilisateur A a confiance en un utilisateur B , qui lui-même accorde sa confiance à C , alors A a confiance (totalement ou partiellement) en C . Notons toutefois que PGP peut être utilisé sans faire usage de la transitivité de la confiance.

2.8 Matériels didactiques et références bibliographiques

- [1] RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, mai 2008. <https://www.ietf.org/rfc/rfc5280.txt>
- [2] AVOINE GILDAS, *Certificates*, Transparents de cours. <http://www.avoine.net/cyberedu/>
- [3] RFC 4880, *OpenPGP Message Format*, novembre 2007, <http://www.ietf.org/rfc/rfc4880.txt>

3 Fiche 3 : La fédération d'identité

3.1 Thématique

Thématique	Authentification	Numéro de fiche	3	Mise à jour	07/03/2016
------------	------------------	-----------------	---	-------------	------------

3.2 Thème des cours visés

Cette fiche peut être utilisée dans un cours traitant de l'administration d'un système (base de données, réseau ou web).

Il peut être intéressant également d'aborder les concepts présentés dans cette fiche dans un cours de systèmes d'exploitation.

Plus généralement, il convient d'utiliser cette fiche dans tout cours traitant ou abordant des sujets relatifs à la création des comptes utilisateurs, l'ajout ou la suppression d'informations relatives à ces comptes et leur maintenance.

3.3 Volume horaire

Il est estimé à 1 heure. Ce volume horaire peut être étendu à 1 heure et 30 minutes si l'enseignant choisit de développer plus avant des exemples. Il est recommandé d'accorder 5 à 10 minutes à chaque classe de concepts traitée dans cette fiche.

3.4 Prérequis / corequis

La notion d'identité numérique est au cœur de la fédération d'identité. Une connaissance minimale des concepts associés est nécessaire notamment le concept d'identifiant numérique (le « *login* » utilisateur), l'intérêt d'en posséder un, et la représentation de l'identité numérique d'un individu à l'aide d'attributs personnalisés (nom, prénom, adresse, âge, etc.).

Les sujets énumérés ci-après sont connexes à la fédération d'identité. Ils peuvent être présentés rapidement par l'enseignant une fois les concepts de cette fiche traités dans le cours ou faire l'hypothèse qu'ils sont étudiés par ailleurs, afin de matérialiser l'intégration d'une fédération d'identité au sein d'un système d'information :

- le *provisioning* (au sens de l'allocation automatique) de comptes ;
- les services d'annuaires utilisateurs : LDAP (*Lightweight Directory Access Protocol*), *Active Directory*, *NDS Novell Directory Services* ;
- les systèmes d'authentification (voir les autres fiches pédagogiques relatives à la thématique) ;
- les systèmes d'*accounting* et d'audit : enregistrement des transactions (qui a fait quoi, quand et sur quelles ressources) ;
- la déconnexion unique (*Single-Log-Out*).

3.5 Objectifs pédagogiques

Bien que cela soit en amont des concepts de la fédération d'identité, le premier objectif pédagogique vise à sensibiliser les étudiants au problème de la gestion de l'identité numérique.

Un second objectif est de leur faire comprendre que la gestion de l'identité ne peut être centrée uniquement sur l'institution d'appartenance. Cette approche ne peut répondre efficacement à des architectures distribuées, à des collaborations avec des tiers en dehors de l'institution d'appartenance ou encore la composition des services offerts. La fédération d'identité permet de répondre à ces besoins.

Il s'agit également d'appréhender les concepts de délégation d'authentification et d'autorisation en lien avec la gestion de l'identité numérique et de sensibiliser aux notions essentielles de « fournisseur d'identité » et de « fournisseur de service » et plus généralement de tiers de confiance.

3.6 Conseils pratiques

Présenter le mécanisme d'authentification unique (*Single Sign-On*) [1]. Comme il est à l'origine du concept développé dans cette fiche, il constitue une brique de base sur laquelle le cours peut s'appuyer. Les étudiants en font fort probablement usage sans forcément en être conscients pour la plupart, donc il est conseillé de donner des exemples (p. ex. la saisie d'un *login*/mot de passe dans un système Windows intégrant la SSO dans *Active Directory* permet ensuite un accès direct à la messagerie).

Rappeler le contexte. Les organisations avaient besoin d'un moyen d'unifier les systèmes d'authentification dans l'entreprise pour une gestion plus facile et une meilleure sécurité. Le SSO a été largement adopté et a fourni une solution pour maintenir un référentiel de noms d'utilisateurs et mots de passe qui pourraient être utilisés de façon transparente à travers plusieurs applications internes. Un des changements qui ont suivi l'introduction de ce mécanisme de SSO *classique* a été de fournir les logiciels comme un service. Il résulte du besoin des organisations d'ouvrir par exemple des interfaces de programmation (API - *Application Programming Interface*) afin que les partenaires et développeurs externes puissent utiliser leurs logiciels. Le défi a été de gérer les authentifications et les autorisations des consommateurs de ces APIs.

Poser le problème auquel la fédération d'identité apporte une solution. Comment simplifier dans ce contexte l'authentification et accroître la sécurité, c-à-d : comment rassembler les informations de connexion des utilisateurs au travers les nombreuses applications et plateformes, comment éviter que chaque application doive gérer un ensemble de *credentials* (attestations de qualification ou de compétence, certificats) pour chaque utilisateur et comment éviter que chaque utilisateur doive se remémorer le *login*/mot de passe à utiliser pour chaque application qu'il sollicite.

3.7 Description

3.7.1 Définitions

L'identité numérique : Une identité numérique représente un ensemble d'informations numériques utilisé par un système informatique pour représenter un utilisateur. La connaissance de ces informations numériques permet à un système informatique de prendre des décisions (d'autorisation d'accès par exemple) de façon autonome lors des interactions avec cet utilisateur.

L'IAM : La gestion des identités et des autorisations (IAM - *Identity and Access Management*) a pour objectif de définir un ensemble de solutions intégrées pour assurer une gestion centralisée et cohérente des identités et des fonctions d'authentification et de contrôle d'accès. Elle comprend quatre fonctions : (1) La gestion des identités numériques, voir définition ci-après, (2) la fédération des

identités, objet de la présente fiche, (3) la définition et le déploiement des politiques d'autorisations et (4) la gestion des autorisations qui contrôle les évolutions de la politique d'autorisation.

La gestion des identités : C'est l'élément de base qui sous-tend le bon déploiement d'une solution d'IAM. Il repose sur une gestion centralisée ou hiérarchique des identités des utilisateurs et de l'authentification. Ses principales fonctions, outre l'authentification (voir les autres fiches portant sur la thématique) :

- **Collecte** : De nombreux composants différents peuvent avoir été utilisés pour gérer les identités de l'entreprise, tels que annuaire LDAP, serveur RADIUS, infrastructure à clé publique (PKI), logiciel de gestion du personnel, logiciel de gestion de comptes (SUN NIS/NIS+, Microsoft *Active Directory*). La fonction de collecte consiste à recenser ces différents composants de manière à constituer un annuaire centralisé des différentes identités. Les mécanismes de fédération d'identités peuvent s'appuyer sur un ou plusieurs annuaires centralisés.
- **Réconciliation** : Une fois l'annuaire centralisé constitué, la réconciliation a pour objectif d'assurer la cohérence de cet annuaire. Il s'agit principalement de résoudre les conflits et éliminer les redondances : chaque individu recensé dans l'annuaire doit avoir une identité unique et cette identité unique est mise en correspondance avec les différents annuaires distribués utilisés par l'entreprise (mise en correspondance par une fonction de *mapping*). Le résultat de la réconciliation est un annuaire centralisé (ou méta annuaire) cohérent qui peut être interrogé par les systèmes d'exploitation et les applications via des protocoles sécurisés (LDAPS par exemple).

3.7.2 La fédération d'identité : présenter le principe de base

Supposons que plusieurs applications ou services sont mis à disposition des utilisateurs. Une fois qu'un utilisateur s'est authentifié auprès de l'un de ces services ou applications, la fédération d'identité permet d'informer les autres services de l'identité de cet utilisateur et valider son authentification au moyen d'un protocole dit de confiance (SAML2 [2], Open ID [3], Shibboleth [4], etc.). Il n'a ainsi plus besoin de s'authentifier à nouveau auprès des autres services liés par une relation de confiance. Ces services ont préalablement constitué pour ce faire un cercle de confiance (voir figure 1).

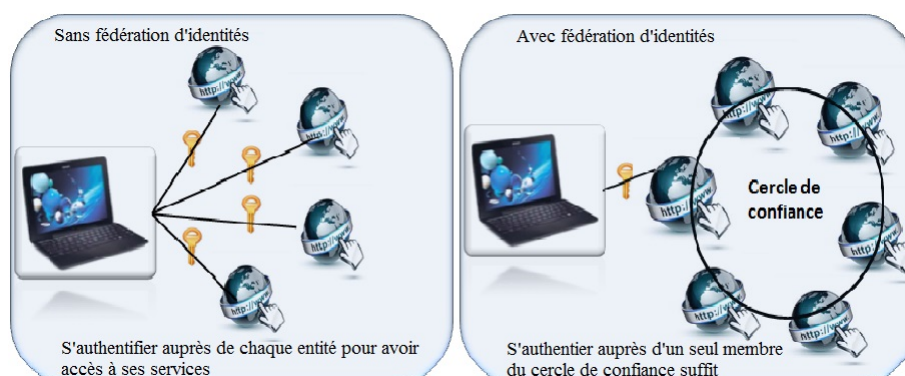


FIGURE 1 – Principe de base de la fédération d'identité.

Formellement, la fédération d'identité sous-entend la création de cercles de confiance (ou CoT — *Circle of Trust*). Un CoT est un ensemble d'entités fournissant des services et des identités (ou des attributs) entre lesquelles une relation de confiance a été établie.

Les entités qui fournissent un service ou une application dans ce CoT protègent l'accès aux applications, bloquent tout accès sans authentification préalable et redirigent l'utilisateur non authentifié

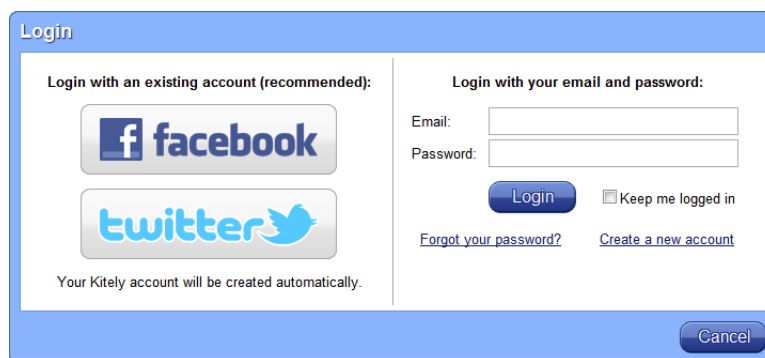


FIGURE 2 – Fournisseurs d'identité.

vers son fournisseur d'identité. Les entités qui fournissent l'identité se chargent d'authentifier l'utilisateur ainsi que de récupérer des informations additionnelles (attributs) sur son identité susceptibles d'être réclamées par les autres entités du CoT avant qu'elles n'autorisent l'accès. L'enseignant peut prendre des exemples familiers aux étudiants pour illustration, p. ex. Twitter et Facebook jouent ce rôle pour l'accès à certaines applications (voir figure 2).

Après une authentification unique auprès d'une de ces entités, l'utilisateur peut accéder librement aux services fournis par ce CoT, de façon sécurisée et la plus transparente possible. L'enseignant peut illustrer cette notion de CoT par un cas d'usage. Un exemple est donné dans la figure 3 ci-après.

3.7.3 La fédération d'identité : présenter les principales fonctions

SSO interne. Il repose sur une base de données globale et centralisée réunissant les données d'identification de tous les utilisateurs (un méta annuaire). La mise en place d'une gestion d'identités (voir section 3.7.1) centralisée et cohérente dans une entreprise souhaitant fédérer les identités dans son système est donc une étape préalable au bon déploiement d'une authentification unique interne.

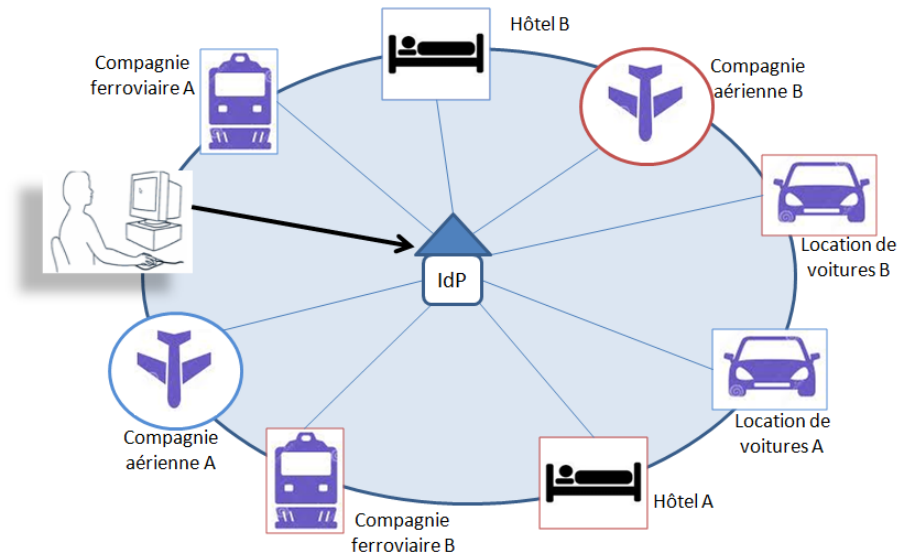
Une autorité gérera l'authentification auprès des différents services et applications en arrière plan sans l'intervention de l'utilisateur. Ce procédé permet de lui éviter de devoir fournir sans cesse des informations d'authentification.

L'enseignant soulignera les points importants suivants :

- L'authentification validée par cette autorité garantit que l'utilisateur est bien celui qu'il prétend être, mais elle n'a pas pour autant valeur de permission d'accès. Les autorisations sont gérées séparément ;
- Comme le SSO permet d'accéder à plusieurs services une fois que l'utilisateur est authentifié, il peut augmenter le risque de compromission des services si le certificat d'authentification présenté par l'utilisateur vient à être utilisé illégalement par d'autres utilisateurs. Par conséquent, le mécanisme de SSO doit reposer sur une protection efficace des secrets d'authentification (voir la fiche portant sur les certificats).

L'enseignant développera plus en avant des solutions existantes. À titre d'exemples :

- *Windows Integrated SSO* : les services associés permettent de se connecter à diverses applications dans le réseau de l'entreprise qui utilisent le même mécanisme d'authentification. Si par exemple, le protocole utilisé est Kerberos, après authentification des *credentials* de l'utilisateur sollicitant une ressource accessible via ce protocole, il peut accéder à toutes celles intégrées avec ce même protocole.



Un portail de voyages en ligne est un bon exemple d'illustration d'un CoT. Il s'agit d'un site web conçu pour aider à trouver un accès à divers fournisseurs de services de voyage à partir d'un seul site. Le portail de voyages forme un partenariat avec des hôtels, des compagnies aériennes, des agences de location de voiture, etc. et les présente sur son site web. L'utilisateur se connecte à ce portail et cherche un hôtel compatible avec ses contraintes. Une fois la réservation d'hôtels terminée, l'utilisateur se déplace vers la partie compagnies aériennes du portail à la recherche d'un vol. En raison de l'accord de partenariat avec le portail de voyages qui joue le rôle de fournisseur d'identité (IdP, Identity Provider), le site de la compagnie aérienne partage les informations d'authentification obtenues plus tôt dans la session en ligne de l'utilisateur. Celui-ci se connectera au site de réservation d'avion sans avoir à se ré-authentifier. Tout ceci est transparent pour l'utilisateur.

FIGURE 3 – Cercle de confiance - Portail de voyage.

- *Server-Based Intranet SSO* : il permet d'intégrer des applications et des systèmes hétérogènes dans l'environnement de l'entreprise n'utilisant pas le même moyen d'authentification (p. ex. *Windows Active Directory* pour l'un et *Resource Access Control Facility* pour l'autre).
- *CAS (Central Authentication Service)* : lorsque le client s'authentifie sur le serveur, il reçoit un ticket sous forme de cookie dans son navigateur et qui lui permettra de passer d'un service à un autre sans devoir s'authentifier à nouveau. Ce cookie ne peut être lu ou écrit que par le serveur CAS. Si le navigateur refuse les cookies, le client sera redirigé vers le serveur CAS à chaque fois qu'il accédera à un service.

SSO externe. Dans les solutions de SSO interne, les utilisateurs sont connus à l'avance par le système. L'autorisation est fondée sur leur identité qui est vérifiée d'une manière centralisée (donc localement), et obtenue après authentification. Comme les architectures actuelles sont de plus en plus distribuées, les organisations ont recours à des mécanismes d'authentification différents, déployés au sein de chaque service. L'authentification unique externe apporte une solution à ce problème et se fonde essentiellement sur une confiance établie entre des organisations et leurs services qui collaborent. Des solutions standardisées ont été définies pour assurer l'interopérabilité dans ce contexte (voir la figure 4).

Le SSO externe repose sur deux mécanismes :

Organisation	Standard
OASIS (<i>Organization for the Advancement of Structured Information Standards</i>)	SAML (<i>Security Assertion Markup Language</i>) : langage pour rendre disponibles des assertions de sécurité et des attributs
Liberty Alliance	ID-FF (<i>Identity Federation Framework</i>) : spécification de base permettant la création d'un réseau de fédération d'identité. FF comporte des schémas, des protocoles et des profils.
	ID-WSF (<i>Identity Web Service Framework</i>) : description d'une structure de base de services d'identité tels que la découverte de ressources.
Shibboleth Project	Un standard de fédération d'identité développé pour une utilisation dans les institutions académiques.
Microsoft, IBM	WS-Federation : il permet l'authentification mutuelle d'applications.

FIGURE 4 – Standards et fédération d'identité.

La délégation de l'authentification. Les mots de passe ne circulent plus sur le réseau et un service (le fournisseur d'identité) s'occupe de centraliser l'identité de l'utilisateur. Le fournisseur d'identité centralise l'ensemble des attributs permettant d'authentifier l'utilisateur. Lorsqu'un utilisateur souhaite utiliser les fonctions d'un service nécessitant une authentification, ce fournisseur de services ne va donc plus l'authentifier directement mais le rediriger vers un fournisseur d'identité. Ce dernier va ensuite authentifier l'utilisateur puis le rediriger vers le service initialement demandé. À cette redirection sera attachée une assertion de sécurité, matérialisée par un message sécurisé, à l'attention du service pour lui indiquer le résultat de l'authentification utilisateur. Il peut s'agir d'un message au format XML avec SAML2 (standard OASIS) ou au format JSON avec JWT (RFC 7519). Cette assertion de sécurité est un message numérique signé électroniquement par un partenaire de confiance préalablement référencé qui permet la réalisation d'une délégation de l'authentification et d'une façon plus générale la mise en place de mécanismes comme l'authentification unique auprès de plusieurs services d'une même fédération ou encore la délégation d'autorisation d'accès et d'usage. La section 3.8 et la figure 5 décrivent plus en détails l'exemple des flux Shibboleth qui peuvent servir d'illustration et que l'enseignant peut développer plus en avant. Il peut également présenter la FIM Microsoft (*Front Identity Manager*) et son composant AD FS 2.0 [5] qui est compatible au standard *WS-Federation* [6].

Pour aller plus loin : une fédération peut contenir plusieurs fournisseurs d'identité. L'utilisateur sera alors redirigé vers le fournisseur d'identité par défaut ou celui qu'il aura précisé au service contacté. La mécanique pour la délégation d'authentification est ensuite similaire à celle précédemment décrite.

À l'issue de la phase d'authentification, le fournisseur de services dispose d'un identifiant de l'utilisateur authentifié auprès de son fournisseur d'identité. Cet identifiant peut ainsi être utilisé par le fournisseur de services pour ensuite obtenir, auprès du fournisseur d'identité, différents attributs sur l'utilisateur. Ces attributs pourront servir à personnaliser le service rendu à l'utilisateur par exemple, tout en gardant la possibilité de ne pas suivre (tracer) l'utilisateur lors d'utilisations successives d'un même service et/ou de ne pas être en mesure de savoir qu'un même utilisateur a consommé plusieurs services (notion d'identifiant opaque unique par accès ou service).

La délégation de l'autorisation. Une fois l'utilisateur authentifié, celui-ci peut être autorisé ou non à utiliser un service donné, voire certaines ressources d'un service donné. Le fournisseur de services peut décider de gérer lui-même ce type d'autorisation mais peut également déléguer cette gestion à un service extérieur. On parle dans ce cas de délégation d'autorisation qui est mise en œuvre à l'aide d'échanges de messages sécurisés entre les différentes parties prenantes de façon similaire aux échanges pour la délégation d'authentification.

3.8 Matériels didactiques et références bibliographiques

La fédération d'identité a vu l'émergence d'un certain nombre de standards. La présentation du fonctionnement d'un de ces standards, comme Shibboleth par exemple, permet de saisir le principe de la fédération d'identité (voir figure 5).

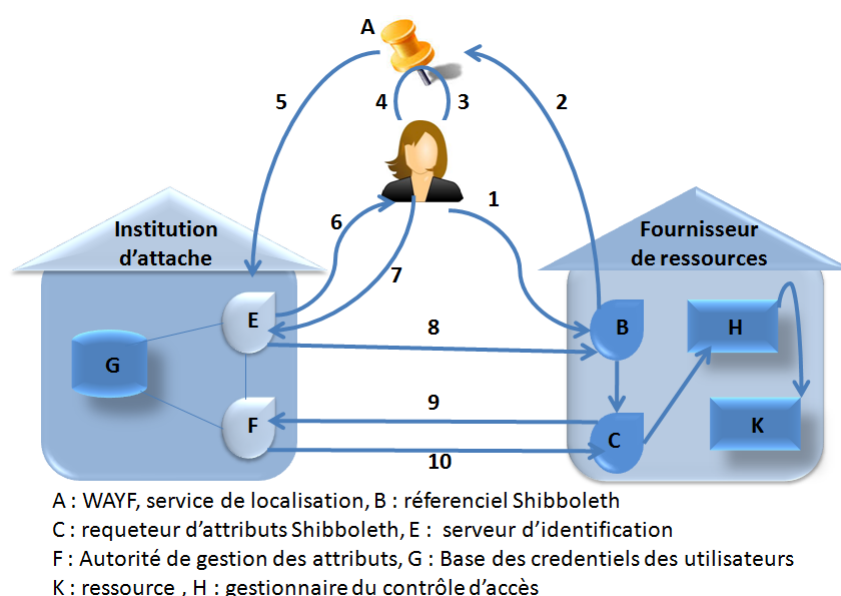


FIGURE 5 – Fonctionnement de Shibboleth.

1. Un utilisateur demande l'accès à une ressource (K) en interrogeant le fournisseur de la ressource en question (B) ;
2. Le fournisseur le renvoie vers un service de localisation (A : WAYF, *Where Are You From*) qui lui demande de quelle institution il provient. Cette action est transparente du point de vue de l'utilisateur ;
3. et 4. Le service de localisation auto-sélectionne l'origine de l'utilisateur ;
5. L'utilisateur est redirigé vers son institution d'attache (E) pour y être identifié et authentifié et ce, avec le même identifiant et authentifiant qu'il utilise habituellement ;
6. et 7. l'authentification de l'utilisateur est effectuée ;
8. Le service Shibboleth de l'institution d'attache, authentifie l'utilisateur auprès du fournisseur de service. Il crée d'abord un numéro de transaction (un descripteur) et le redirige vers le fournisseur de service (B) ;
9. et 10. En arrière plan le fournisseur de service valide le descripteur et échange des attributs avec l'institution d'attache.

Enfin l'accès est accordé conformément au profil de l'utilisateur et aux règles d'utilisation.

Voici quelques exemples de fédération Shibboleth : InCommon (Internet2), SWITCH (*The Swiss Education*) [7], HAKA (*Finnish IT center for science*) [8].

Remarque : Eduroam n'est pas à base de fédération d'identité. Mais, une institution peut à la fois utiliser (1) Eduroam pour l'accès Wi-Fi nomade inter établissements et (2) la fédération Education-Recherche pour l'accès à des ressources numériques en ligne situées en dehors du périmètre de son système d'information. La figure 6 illustre le fonctionnement d'Eduroam.

Ce qu'il faut retenir c'est que Eduroam permet l'itinérance dans les milieux académiques. Il vise à offrir un accès sans fil sécurisé à Internet aux personnels des établissements d'enseignement supérieur et de recherche lors de leurs déplacements en utilisant leur mot de passe habituel.

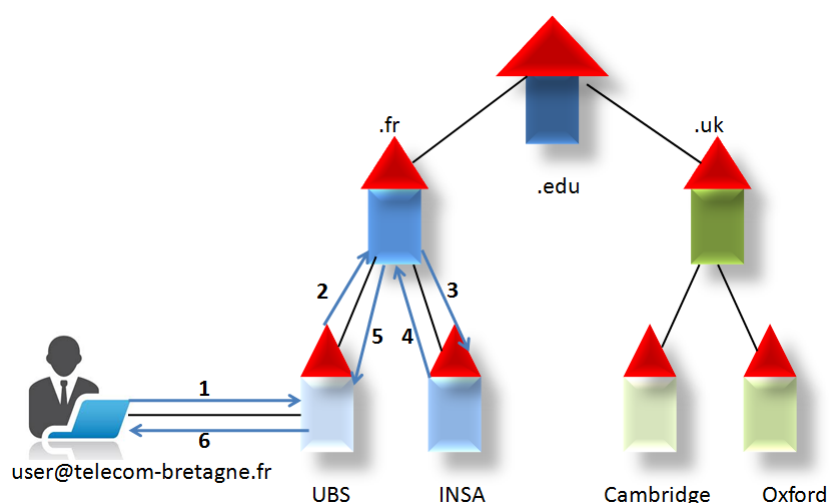


FIGURE 6 – Eduroam : exemple d'illustration.

- [1] SANS Institute, InfoSec Reading Room, *Secure implementation of Enterprise single sign-on product in an organization*, juillet, 2014. <https://www.sans.org/reading-room/whitepapers/authentication/secure-implementation-enterprise-single-sign-on-product-organization-1520>
- [2] Security Assertion Markup Language (SAML) V2.0. Technical Overview, mars, 2008. <https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-discretionary{-}{-}{-}overview-2.0-cd-02.pdf>
- [3] OpenID Foundation. <http://openid.net/>
- [4] Shibboleth. <https://shibboleth.net/>
- [5] Web Services Federation Language (WS-Federation), version 1.1, décembre 2006. BEA, BMC Software, CA Inc., IBM, Layer 7 Technologies, Microsoft, Novell Inc., VeriSign.
- [6] Understanding WS-Federation, mai 2007, version 1.0. IBM, Microsoft.
- [7] The Swiss Education & Research Network, <http://www.switch.ch/aai/demo/>
- [8] Le réseau fédéré finlandais, <https://haka.funet.fi/shibboleth/>

4 Fiche 4 : Protocoles d'authentification et d'échange de clé

4.1 Thématique

Thématique	Protocoles d'authentification et d'échange de clé	Numéro de fiche	4	Mise à jour	07/03/2016
-------------------	---	------------------------	---	--------------------	------------

4.2 Thème des cours visés

Cette fiche sur les protocoles d'authentification et d'échange de clé peut s'inscrire dans un cours d'administration des réseaux ou de développement d'applications.

4.3 Volume horaire

- 1h30 minutes pour présenter les différentes étapes de la construction d'un protocole d'authentification et d'échange de clés.
- 30 minutes pour aller plus loin.

4.4 Prérequis / corequis

Comprendre le principe général des techniques de chiffrement à clé publique et de signature.

Comprendre le principe général des techniques de chiffrement symétrique et de fonction de hachage.

4.5 Objectifs pédagogiques

L'objectif de ce cours est d'expliquer le fonctionnement des protocoles d'authentification et d'échange de clés qui permettent à plusieurs entités de s'authentifier et de générer des clés communes. On verra que la sécurité d'un tel protocole doit lutter contre le rejeu, les attaques par le milieu et l'usurpation d'identité, et garantir la *forward secrecy*.

4.6 Conseils pratiques

Dans ce cours, on part du protocole d'échange de clé bien connu Diffie-Hellman et on va montrer comment le rendre sécurisé en contrant certaines attaques dont l'attaque par le milieu, le rejeu et l'usurpation d'identité. À la fin, on verra la propriété de *forward secrecy* et pourquoi la dernière étape de diversification des clés est essentielle à la sécurité. Le protocole que l'on construit petit à petit est proche du protocole IKE, utilisé dans IPsec.

Il peut être intéressant de compléter cette fiche par un TP sur le protocole Kerberos. En effet, ce dernier utilise de la cryptographie symétrique, et peut être rencontré par les étudiants en administration des réseaux [3].

4.7 Description

Un protocole d'authentification permet à un utilisateur, qu'on appellera *prouveur* ou *client*, de prouver son identité à un participant qu'on appellera *vérifieur* ou *serveur*. Afin de sécuriser le canal de communication utilisé ensuite entre le prouveur et le vérifieur, il est nécessaire que les deux participants s'accordent sur une clé secrète commune qui leur permettra d'authentifier à l'aide des méthodes décrites dans la fiche 5 (et éventuellement de chiffrer) les messages échangés sur ce canal. Le protocole permettant de générer une telle clé secrète est appelé *échange de clé* (ou encore *établissement de clé* ou *négociation de clé*).

Protocoles défi/réponse pour l'authentification

Dans le cas d'un protocole défi/réponse (*challenge/response* en anglais), le serveur envoie un défi et le client doit générer une signature portant sur ce défi avec sa clé privée. En supposant que le serveur a connaissance de la clé publique du client, il pourra vérifier la preuve apportée par le client. En effet, comme la propriété de sécurité d'un schéma de signature garantit que seul le possesseur légitime de la clé privée est capable de générer une signature valide, le vérifieur sera convaincu que la personne qui cherche à s'authentifier est bien la bonne. L'association entre l'identité du client et sa clé publique peut être garantie par une autorité de certification (voir fiche 2). D'autres mécanismes que des signatures, comme des codes d'authentification de messages peuvent être utilisés.

Pour que ce schéma soit sûr, il faut que l'espace des défis soit suffisamment grand pour éviter que le client puisse simplement rejouer une authentification précédente si le même défi est utilisé. Le *rejeu* est la principale attaque contre ce type de protocole et consiste de la part de l'adversaire à envoyer une réponse qu'il a précédemment vue. Afin d'offrir un niveau de sécurité suffisant, il faut prendre en compte le paradoxe des anniversaires pour estimer la taille de l'espace des défis. On peut aussi utiliser des *nonces* (valeur qui ne se répète pas) comme défi, mais cela demande de conserver un état du côté du serveur pour chaque client, ou utiliser le temps avec une horloge.

Enfin, on peut aussi mentionner l'authentification par défi/réponse utilisant un mot de passe et une fonction de hachage. C'est le principe de l'authentification HTTP digest [2] alors que l'authentification par certificat est normalisée dans la norme X509 [1].

Protocoles d'échange de clé authentifiés

Ces protocoles permettent aux participants (clients et serveurs) de s'authentifier et de générer des clés de chiffrement et d'authentification. Ces clés seront ensuite utilisées pour protéger le canal de communication en confidentialité (si besoin) et en intégrité. On parle alors de *canal sécurisé* quand la confidentialité, l'authenticité et l'anti-rejeu des données sont garantis. C'est le cas des protocoles de sécurité comme IPsec⁴, TLS et SSH. La première étape de ces protocoles consiste à authentifier un participant (ou les deux) et à générer des clés pour la seconde étape, c'est-à-dire le canal sécurisé. Dans la suite, nous allons décrire petit à petit les propriétés d'un protocole d'échange de clé authentifié comme celui utilisé dans IKE [4, 5].

Protocole d'échange de clé Diffie-Hellman. Diffie et Hellman ont été les premiers à proposer une solution au problème de l'échange de clé : c'est-à-dire un moyen pour que deux entités se mettent d'accord sur une clé secrète commune en échangeant des messages sur un canal écouté par un adversaire. Ils ont conçu un protocole en deux passes dont la sécurité repose sur un problème de

4. Pour être précis, dans le cas d'IPsec, l'authentification et l'échange de clés sont gérés par le protocole IKE.

théorie des nombres appelé le problème du logarithme discret. Dans ce protocole décrit figure 7, Alice et Bob se mettent d'accord au préalable sur un groupe cyclique G et un générateur g de ce groupe. Alice envoie g^a où a est un élément secret connu d'elle seule et Bob envoie g^b où b est un élément secret connu de lui seul. À la fin, Alice et Bob peuvent calculer la clé g^{ab} mais un adversaire qui écoute le canal doit calculer g^{ab} à partir de g^a et g^b . Ce problème est appelé le problème Diffie-Hellman en cryptographie et c'est un problème algorithmique très difficile dans certains groupes bien choisis.

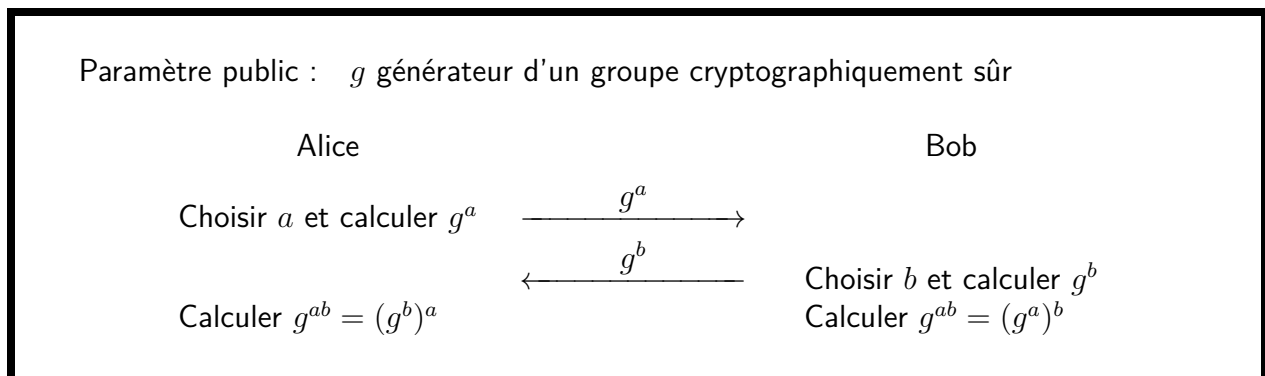


FIGURE 7 – Protocole Diffie-Hellman non-authentifié.

Attaque par le milieu. Ce protocole est vulnérable à une attaque bien connue, appelée *attaque par le milieu* : un adversaire, Charlie, peut échanger une clé avec chacun des deux participants alors que ces derniers croient communiquer entre eux. Charlie est un attaquant actif (car il injecte des messages sur le canal de communication) et peut jouer le rôle de Bob vis-à-vis d'Alice et le rôle d'Alice vis-à-vis de Bob car rien dans les messages ne dépend d'Alice ou de Bob. Le schéma de cette attaque est décrit dans la figure 8. Charlie possède une clé secrète $g^{ab'}$ avec Alice et une autre clé $g^{a'b}$ avec Bob, alors que Alice et Bob croient avoir une clé commune g^{ab} . Charlie pourra déchiffrer tous les messages d'Alice et les renvoyer à Bob et vice-versa sans que les deux parties ne s'en aperçoivent. La confidentialité et l'authenticité des messages est donc compromise.

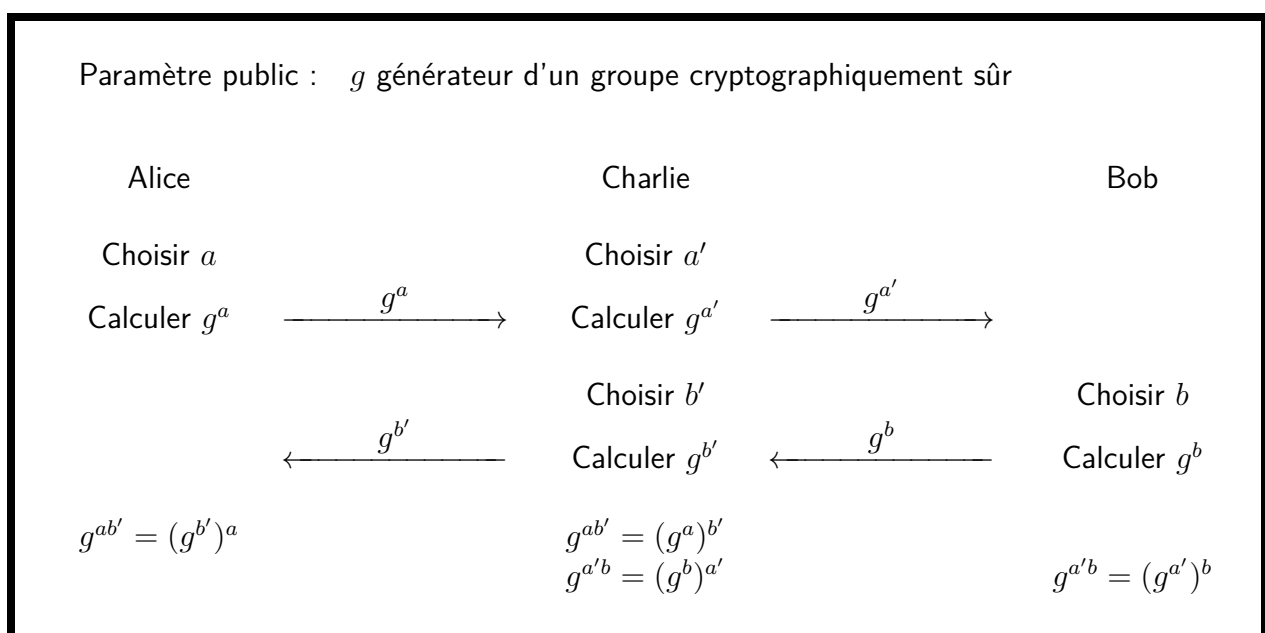


FIGURE 8 – Attaque par le milieu contre le protocole Diffie-Hellman non-authentifié.

Afin de se protéger contre l'attaque par le milieu, il faut authentifier les participants : Bob veut être sûr que le message g^a provient bien d'Alice et de même Alice veut être sûre que g^b provient bien de Bob. On peut utiliser un schéma de signature permettant aux participants de vérifier que c'est bien la bonne personne qui a créé le message. Par exemple, Alice peut envoyer $g^a, \text{Sig}_A(g^a)$ et Bob peut envoyer $g^b, \text{Sig}_B(g^b)$ où Sig est un schéma de signature, garantissant l'authenticité des messages (cf. figure 9).

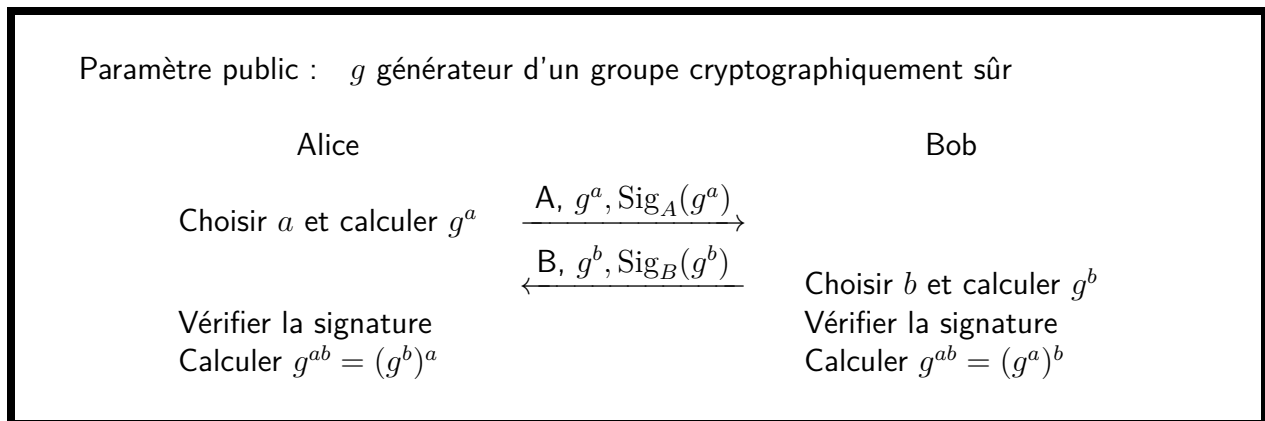


FIGURE 9 – Protocole Diffie-Hellman authentifié, vulnérable au rejeu.

Attaque par rejeu. Malheureusement, le protocole de la figure 9 ne protège pas contre le rejeu car rien n'empêche l'adversaire Charlie de rejouer le message $A, g^a, \text{Sig}_A(g^a)$ vers Bob. À l'issue du protocole, Charlie est parvenu à se faire passer pour Alice (même s'il n'est pas capable de calculer la clé g^{ab} dérivée par Bob). Pour se protéger du rejeu, on peut utiliser un défi et les éléments g^a et g^b sont de bons candidats pour jouer ce rôle. Chacun devra non seulement signer son élément (pour s'authentifier) mais également celui de l'autre (pour éviter le rejeu). On aboutit alors au protocole Diffie-Hellman authentifié en 3 passes décrit dans la figure 10.

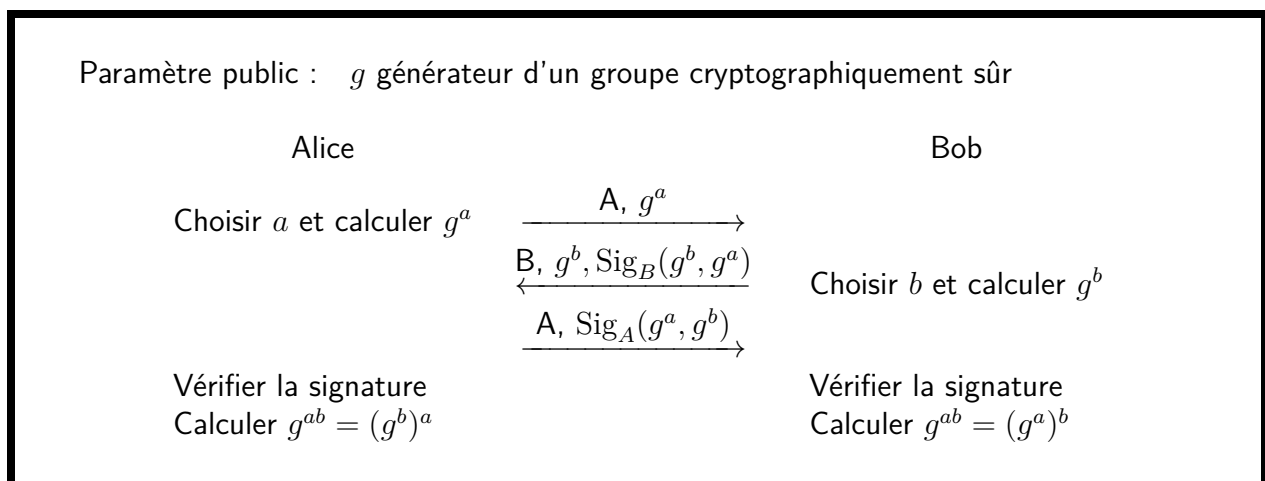


FIGURE 10 – Protocole Diffie-Hellman authentifié vulnérable à l'attaque UKS.

Attaque UKS. Cependant, le protocole de la figure 10 est vulnérable à une attaque subtile décrite dans la figure 11 dans laquelle Charlie remplace l'identité et la signature d'Alice par sa propre identité et sa propre signature (sans modifier les messages eux-mêmes). À l'issue de l'attaque, Alice croit partager une clé avec Bob alors que Bob croit partager une clé avec Charlie. Remarquons que Charlie ne connaît pas la clé g^{ab} ; c'est pourquoi cette attaque est appelée *Unknown Key Share* (UKS). Même si la confidentialité des messages échangés n'est pas compromise, cette attaque peut

avoir de graves conséquences. En particulier, du point de vue de Bob, tous les messages envoyés par Alice « semblent » provenir de Charlie. Dans le cas où Bob est un établissement bancaire par exemple, on pourrait imaginer que les mauvais comptes bancaires soient crédités.

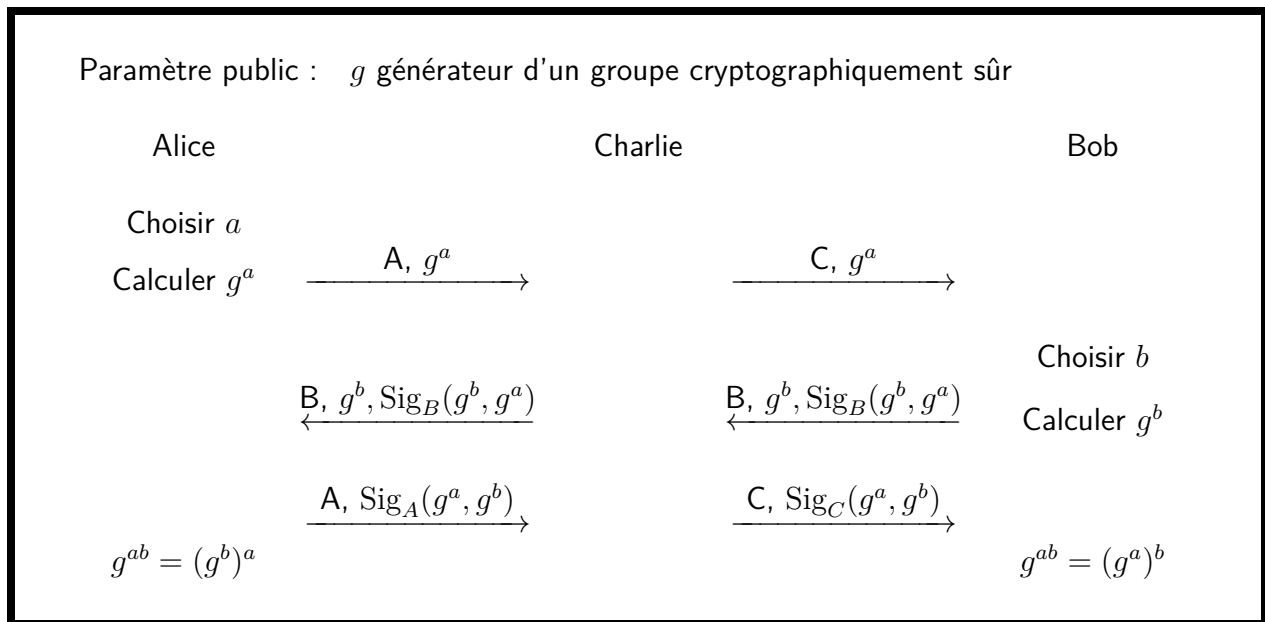


FIGURE 11 – Attaque UKS.

Pour empêcher l'attaque UKS, il suffit d'ajouter les identités des deux participants légitimes dans les messages signés. On aboutit alors au protocole représenté sur la figure 12. Charlie ne peut plus effectuer l'attaque UKS car la signature calculée par Bob sera $\text{Sig}_B(C, B, g^b, g^a)$, alors qu'Alice s'attendrait à recevoir la signature $\text{Sig}_B(A, B, g^b, g^a)$. La vérification de la signature par Alice va donc échouer, et Alice interrompra l'échange. Ce protocole possède une preuve de sécurité.

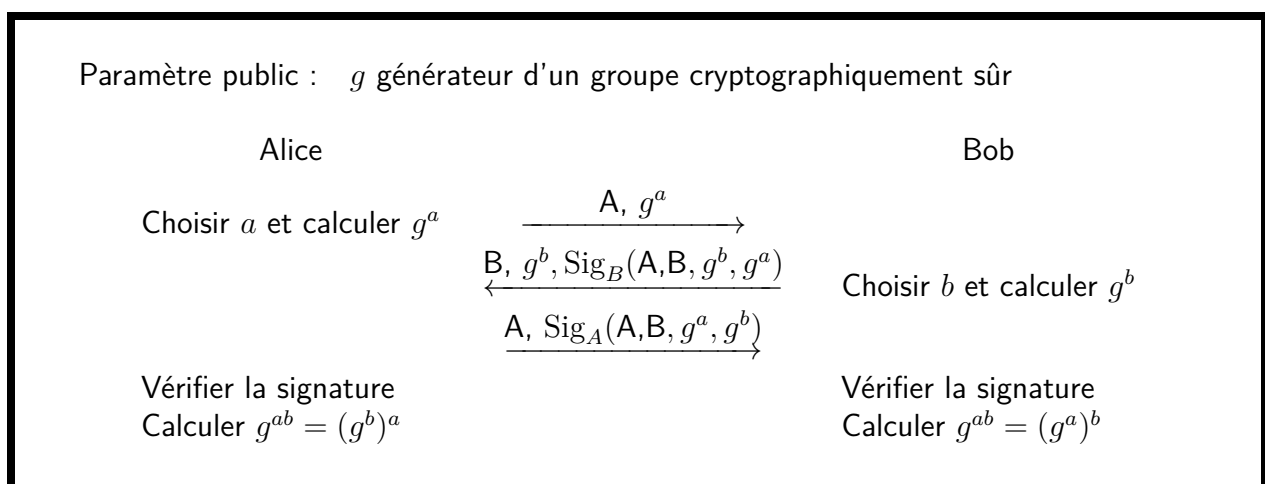


FIGURE 12 – Protocole Diffie-Hellman authentifié sûr.

Forward secrecy. La propriété de *forward secrecy* (parfois appelée confidentialité persistante en français) garantit que même si un attaquant obtient la clé à long terme d'un des participants (c'est-à-dire la clé privée de signature dans le cas précédent), alors les messages que le participant aura envoyés avant cette compromission de son système ne pourront pas être déchiffrés par l'adversaire. Cependant, rien ne pourra garantir la sécurité des messages futurs car l'attaquant qui a compromis la clé de signature du participant pourra se faire passer pour lui. Le protocole Diffie-Hellman authentifié

permet de garantir cette propriété si les participants prennent soin d'effacer les clés éphémères a et b une fois le secret g^{ab} calculé. Il existe d'autres schémas moins utilisés ayant cette propriété [8].

Afin de montrer que tous les mécanismes d'échange de clé ne satisfont pas la propriété de *forward secrecy*, mentionnons le mécanisme d'échange de clé utilisant le chiffrement à clé publique RSA. Dans ce cas, le client génère une clé aléatoire et la transmet chiffrée avec la clé publique du serveur (pour cette raison, cette technique est parfois appelée *transport de clé*). Le serveur peut donc obtenir la clé aléatoire en déchiffrant le message. Cependant, si un adversaire obtient la clé privée RSA du serveur, il peut lui aussi déchiffrer toutes les communications, y compris celles qu'il aurait préalablement enregistrées. C'est une des raisons pour lesquelles le mécanisme d'échange de clé RSA, qui était une option possible jusqu'à TLS 1.2, n'est plus disponible dans TLS 1.3, dans lequel ne subsistent que des variantes du protocole Diffie-Hellman.

Dérivation de clés à partir du secret commun. Le mécanisme Diffie-Hellman permet à Alice et Bob de posséder un secret commun g^{ab} inconnu de l'adversaire. Cependant, rien ne permet d'affirmer que les 128 premiers bits sont uniformément distribués et peuvent être utilisés comme clé de chiffrement ou d'authentification de messages. Un mécanisme de *dérivation de clé*, permettant de convertir le secret commun g^{ab} en clés uniformément distribuées, doit donc être utilisé. Un tel mécanisme peut aisément être construit à partir d'une fonction de hachage par exemple.

Pour aller plus loin.

Authentification par mot de passe. Dans le cas où l'authentification repose sur un mot de passe, le principal problème est d'empêcher les attaques *par dictionnaire* (aussi appelées attaques *par recherche exhaustive*) consistant pour l'attaquant à tester tous les mots de passe un par un. Pour cela, il ne doit pas exister de *test d'arrêt* permettant à l'attaquant de déterminer, à partir d'un seul échange intercepté entre Alice et Bob, si un mot de passe est correct ou non. Ceci permettrait alors une attaque dite *hors-ligne*, c'est-à-dire ne nécessitant pas d'interagir avec l'un des participants légitimes. Lorsqu'il n'existe aucun test d'arrêt, les seules attaques possibles sont les attaques dites par opposition *en ligne* : l'attaquant ne peut alors tester un mot de passe qu'en interagissant avec l'un des participants légitimes. De telles attaques peuvent alors être rendues inopérantes par une politique de sécurité consistant à bloquer un interlocuteur après un faible nombre d'échecs du protocole. Un exemple de protocole d'échange de clé authentifié par mot de passe (et possédant une preuve de sécurité) peut être trouvé dans [7].

Authentification anonyme. Un protocole d'authentification anonyme permet à une personne de prouver qu'elle appartient à un groupe de personnes sans révéler son identité [10]. Ceci garantit une propriété de *privacy* pour les utilisateurs d'un tel système.

4.8 Matériels didactiques et références bibliographiques

- [1] RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, mai 2008. <https://www.ietf.org/rfc/rfc5280.txt>
- [2] IETF, *Hypertext Transfer Protocol (HTTP/1.1) : Authentication (RFC 7235)*, <https://tools.ietf.org/html/rfc7235>
- [3] BORDES AURÉLIEN *Kerberos*, http://www.ssi.gouv.fr/uploads/IMG/pdf/Aurelien_Bordes_-_Secrets_d_authentification_episode_II_Kerberos_contre-attaque.pdf

- [4] KRAWCZYK HUGO, *SIGMA : The 'SIGn-and-MAC' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols*, <http://www.di.ens.fr/~fouque/ipsec.pdf>
- [5] BRESSON EMMANUEL, *Authentification et Echange de clé*, http://www.di.ens.fr/~bresson/P12-M1/P12-M1-Crypto_6.pdf
- [6] BRESSON EMMANUEL, *Identification et Zero-Knowledge*, http://www.di.ens.fr/~bresson/P12-M1/P12-M1-Crypto_9.pdf
- [7] BELLARE MIHIR, POINTCHEVAL DAVID ET ROGAWAY PHILIP, *Authenticated Key Exchange Secure Against Dictionary Attacks*, http://www.di.ens.fr/users/pointche/Documents/Papers/2000_eurocrypt.pdf
- [8] KUNZ-JACQUES SÉBASTIEN ET POINTCHEVAL DAVID, *About the Security of MTI/CO and MQV*, http://www.di.ens.fr/users/pointche/Documents/Papers/2006_scnA.pdf
- [9] POINTCHEVAL DAVID, *Les Preuves de Connaissances et leur Preuves de Sécurité*, Thèse Université de Caen, page 27-39, http://www.di.ens.fr/users/pointche/Documents/Reports/1996_PhDThesis.pdf
- [10] BONEH DAN ET FRANKLIN MATT, *Anonymous authentication with subset queries*, http://www2.parc.com/csl/projects/quicksilver/SDS_papers/Cryptographic_Protocols/annonauth.pdf

5 Fiche 5 : Authentification de messages

5.1 Thématique

Thématique	Authentification de messages	Numéro de fiche	5	Mise à jour	07/03/2016
-------------------	------------------------------	------------------------	---	--------------------	------------

5.2 Thème des cours visés

Cette unité d'enseignement présente le concept d'authentification de messages et peut s'inscrire dans le cadre d'un cours réseau.

5.3 Volume horaire

Un volume horaire d'une heure et demie est suffisant pour couvrir le matériel du cours.

5.4 Prérequis / corequis

Des connaissances de base sur l'authentification sont requises. En particulier, afin de pouvoir contraster ce qui est présenté dans cette unité d'enseignement avec d'autres aspects de l'authentification, l'idéal serait que les étudiants aient étudié comment on peut authentifier un individu (par exemple dans la fiche 1 sur l'authentification par mots de passe).

La connaissance des propriétés d'une fonction de hachage est essentielle pour réussir à comprendre le fonctionnement des codes d'authentification de type HMAC.

La connaissance du principe de fonctionnement d'un chiffrement par blocs est importante pour comprendre le fonctionnement des codes d'authentification de type CBC-MAC.

5.5 Objectifs pédagogiques

Les objectifs pédagogiques principaux du cours sont les suivants :

- introduire le contexte de l'authentification de messages ;
- expliquer la différence entre authentification et confidentialité, donner des exemples de situations où il est possible d'avoir une des deux propriétés sans l'autre (et vice-versa) et insister sur leur complémentarité ;
- présenter de manière abstraite le concept de code d'authentification de messages puis présenter plusieurs implémentations pratiques de cette primitive.

5.6 Conseils pratiques

Après avoir présenté le concept de code d'authentification de messages de manière abstraite, l'enseignant devrait présenter une des deux instanciations pratiques de ces codes mentionnées dans la fiche, soit le HMAC ou le CBC-MAC. Le choix de l'instance à présenter devrait dépendre des primitives avec lesquelles les étudiants sont déjà familiarisés (les fonctions de hachage ou le chiffrement

par blocs). Une fois une de ces constructions présentées, il est conseillé à l'enseignant d'illustrer le problème du rejeu à partir de cette construction.

5.7 Description

Problématique de l'authentification de messages. Le scénario typique considéré ici est celui de l'échange de messages⁵ entre deux participants. Il s'agit d'un contexte différent de celui d'une authentification de type « client-serveur » où un client cherche à convaincre un serveur distant (par exemple un serveur de courriels) de son identité avant d'accéder aux ressources autorisées⁶.

Pour des raisons de fluidité narrative, on appellera ces deux participants Alice et Bob (plutôt que respectivement participant A et participant B). Alice et Bob sont physiquement séparés, mais peuvent échanger des messages grâce à un canal de communication (p. ex. Internet ou le réseau téléphonique). Le problème est que ce canal de communication est aussi sous le contrôle d'un attaquant, qu'on nommera Charlie.

En particulier, on suppose que Charlie peut lire et modifier les messages qu'il voit passer ou injecter de faux messages. Dans ce contexte, l'objectif principal d'Alice et Bob est de pouvoir vérifier l'origine d'un message reçu ainsi que son intégrité. Plus précisément, lorsque Bob reçoit un message censé provenir d'Alice, il doit pouvoir s'assurer que ce message a effectivement été créé par Alice et que son contenu n'a pas été modifié par Charlie (et vice-versa).

L'authentification de messages est une propriété de sécurité qui est complémentaire à la confidentialité, mais qui n'est pas impliquée par celle-ci. Autrement dit, lorsqu'on chiffre une donnée pour en assurer la confidentialité cela ne garantit en aucun cas l'authenticité du message. Par exemple, si on considère un mécanisme de chiffrement comme le masque jetable (*one-time pad* en anglais) celui-ci assure une confidentialité inconditionnelle au sens de la théorie de l'information [3]. Autrement dit, même si Charlie dispose d'une puissance de calcul infinie, il ne pourra jamais réussir à apprendre le contenu du message mieux qu'en essayant de deviner celui-ci au hasard parmi l'espace des messages possibles. Cependant, utilisé de manière isolée, le masque jetable n'assure aucune forme d'authenticité. En particulier, Charlie peut inverser un bit du message chiffré qu'Alice a envoyé à Bob (même sans connaître la valeur de celui-ci). Ainsi, lors du déchiffrement, Bob récupérera à la position modifiée par Charlie un bit dont la valeur sera le complément de la valeur du bit initial.

À l'inverse, il existe des situations où le fait que le contenu du message soit connu de l'adversaire n'est pas problématique, mais pouvoir vérifier son authenticité est la propriété clé qu'on souhaite pouvoir fournir. Par exemple, lorsque le boîtier connecté se trouvant dans une voiture discute avec le système d'ouverture automatique se trouvant sur une porte de garage, l'essentiel pour la porte est de pouvoir s'assurer de l'authenticité et l'intégrité du message d'ouverture. Autre exemple plus critique, lorsqu'un serveur bancaire reçoit l'ordre d'effectuer une certaine transaction, il est essentiel qu'il puisse vérifier l'authenticité de cet ordre. Pour ces situations-là, les codes d'authentification de messages qui sont décrits un peu plus loin permettent (comme leur nom l'indique) d'authentifier un message mais n'offrent aucune protection concernant sa confidentialité. En général, on parle de « canal sécurisé » lorsqu'on utilise un protocole permettant d'assurer à la fois la confidentialité et l'authenticité des messages.

5. De manière générale le terme « message » peut se référer ici à n'importe quel type de donnée.

6. Cependant, les primitives développées pour répondre à l'authentification de messages sont souvent utilisées comme briques de base dans d'autres contextes, incluant l'authentification de type client-serveur.

Codes d'authentification de messages. Tout d'abord, il est important de préciser que les mécanismes qui sont conçus pour détecter des erreurs générées aléatoirement par un bruit sur le canal de communication ne fournissent aucune sécurité en présence d'un adversaire actif qui peut modifier les messages envoyés ou injecter de faux messages sur le canal de communication. Par exemple, si Alice envoie à Bob simplement un message m et son haché $h(m)$, cela n'offre aucune garantie sur l'authenticité du message. En particulier, Charlie pourrait très bien générer aussi un message arbitraire de son choix m' et le haché correspondant $h(m')$. Lorsqu'il reçoit un message, Bob n'a a priori aucun moyen de distinguer entre un message qui vient réellement d'Alice ou un message généré par Charlie qui semble provenir d'Alice.

Une première manière de réaliser l'authentification de messages est de se placer dans le modèle symétrique où on présuppose qu'Alice et Bob partagent une clé secrète k qui est inconnue de Charlie. Ce secret k peut par exemple avoir été généré lors d'une précédente interaction entre Alice et Bob ou encore être obtenu par un canal auxiliaire sur lequel Charlie n'a aucun contrôle. Dans ce modèle symétrique, Alice et Bob peuvent utiliser un code d'authentification de messages se basant sur k pour être capable de vérifier l'authenticité des messages échangés. De manière abstraite, un code d'authentification de messages est une construction qui comprend deux algorithmes :

1. Un *algorithme de génération de code* qui prend en entrée un message m et la clé secrète partagée k et produit en sortie le code d'authentification de ce message $s = CAM_k(m)$.
2. Un *algorithme de vérification de code* qui prend en entrée un message m , la clé secrète partagée k et un code d'authentification s et qui fournit une réponse binaire en sortie suivant que le code est accepté ou non. En général, l'algorithme de vérification fonctionne en recalculant le code d'authentification à partir du message m et de la clé k et en vérifiant que le code généré est bien égal au code s qui lui a été transmis.

En théorie, il existe des mécanismes d'authentification de messages offrant une sécurité inconditionnelle. En pratique, ils ne sont quasiment jamais utilisés à cause de l'espace mémoire requis. Par exemple, Alice et Bob pourraient lors d'une rencontre passée avoir créé un tableau dans lequel ils énumèrent à l'avance tous les messages possibles qu'ils pourraient un jour vouloir se communiquer. Ensuite, ils vont associer à chacun de ces messages une chaîne de bits de taille κ qui est générée de manière aléatoire et indépendante des autres chaînes et qui jouera le rôle de code d'authentification. Plus tard, si Alice veut envoyer un message à Bob, elle n'a qu'à chercher dans le tableau le code correspondant et à l'envoyer en même temps que son message. Si on fait l'hypothèse additionnelle que Charlie ne peut pas empêcher le message de se rendre à destination, Bob pourra vérifier l'authenticité de ce message en allant vérifier le code correspondant dans le tableau. Par la suite, il est important que Bob n'accepte par contre plus jamais ce message, car il pourra avoir été rejoué par Charlie hors de son contexte originel. Dans ce code d'authentification de messages, la meilleure chose que Charlie puisse faire pour générer le code correspondant à un faux message m' est de tenter de deviner le code associé. Bob considérera le message m' comme étant authentique avec une probabilité de l'ordre de $\frac{1}{2^\kappa}$ ce qui peut être considéré comme négligeable si on choisit κ suffisamment grand (p. ex. de taille 128 ou 256 bits). De plus comme chaque code est généré de manière indépendante, il est clair que Charlie ne gagne aucune information sur les codes de messages inconnus à partir de ceux qu'il voit passer sur le canal de communication. Cependant, le problème de cette méthode est qu'elle est très gourmande en terme de mémoire utilisée. Par exemple, si on considère un tableau contenant tous les messages écrits sur 1 000 bits qu'Alice et Bob pourront un jour vouloir s'envoyer alors même si on essayait d'écrire chaque ligne du tableau sur chacun des atomes de l'univers, il n'y aurait pas assez d'atomes existants pour cela. Il est donc nécessaire de s'intéresser à des codes d'authentification de messages offrant de bonnes propriétés de sécurité, mais se basant sur des clés courtes et réutilisables.

HMAC. Une des manières les plus standards d'implémenter un code d'authentification est une construction se basant sur la combinaison d'une fonction de hachage avec une clé secrète connue sous le nom de HMAC [1] (*keyed-Hash Message Authentication Code*). Dans cette construction, l'algorithme de génération du code consiste à appliquer une fonction de hachage cryptographique h sur une entrée obtenue à partir du message m et de la clé secrète k . La construction HMAC repose essentiellement sur deux appels à la fonction de hachage combinant m et k (la spécification exacte fait également intervenir deux chaînes de caractères fixes) : $CAM_k(m) = h(k || h(k || m))$. Sans ce deuxième appel, la construction ne serait pas sûre avec des fonctions de hachage construites de manière itérative, telles que SHA-256. Pour vérifier le HMAC, il suffit de recalculer le motif d'intégrité attendu avec k et m de la même manière, et de n'accepter le message comme authentique que si l'empreinte générée correspond au code transmis sur le canal de communication.

CBC-MAC. Une autre manière standard de créer un code d'authentification de message est de se baser sur un algorithme de chiffrement par blocs (p. ex. DES ou AES). Ainsi, si on utilise le mode de fonctionnement par enchaînement de blocs (*Cipher Block Chaining* ou CBC en anglais), le dernier bloc chiffré sera fonction de tous les blocs précédents et pourra être utilisé comme code d'authentification de messages car dépendant de tous les blocs de celui-ci. Plus précisément, on commence d'abord par découper le message m en blocs de taille constante ($m = m_1, m_2, \dots, m_l$), puis on calcule itérativement $c_i = E_k(c_{i-1} \oplus m_i)$, où c_0 est un vecteur d'initialisation fixé et connu ; le code d'authentification est alors la valeur finale c_l . Ainsi, une relation d'interdépendance se crée entre les chiffrements des différents blocs, et la valeur du dernier bloc chiffré est dépendante des valeurs de tous les blocs du message. On peut donc utiliser ce dernier bloc comme un code d'authentification de messages qui est fonction de m et de k .

La construction d'un code d'authentification de messages se basant sur un chiffrement par blocs en mode CBC (abrégé CBC-MAC) a été prouvée comme sûre sous les conditions suivantes [2] :

- le chiffrement par blocs utilisé est sûr ;
- la longueur des messages à authentifier est fixe et connue.

La sécurité du mode CBC-MAC devient cependant plus problématique lorsque la taille du message peut varier, auquel cas il est crucial de faire attention à la manière de combler un message pour que la taille d'un message atteigne un multiple de la taille des blocs. Lors de cette étape, appelée *padding* (ou bourrage en français), il faut garantir qu'aucun message complété ne puisse être le préfixe d'un autre message complété. Dans le cas contraire, un attaquant pourrait déduire un code d'authentification pour un message m' à partir d'un code d'authentification pour un message m , sans connaître la clé. Une des manières d'atteindre cette propriété est de préciser la taille du message au début de celui-ci en plus de rajouter des zéros à fin du message jusqu'à ce qu'il atteigne une taille multiple de la taille de bloc. Une autre méthode courante pour rendre CBC-MAC sûr dans le cas de messages de taille quelconque consiste à sur-chiffrer la valeur finale de CBC-MAC avec une clé indépendante de la clé utilisée dans le chaînage.

La question du rejeu. L'intégrité n'implique pas qu'un message est récent. En effet, il n'y a aucune notion de temporalité dans un MAC. Par conséquent, une attaque par rejeu est possible si Charlie arrive à renvoyer un message observé par le passé hors de son contexte original. Supposons par exemple qu'Alice envoie le message suivant à Bob lundi « rendez-vous au restaurant ce soir à 20h » avec le code d'authentification de messages associé. Charlie pourra ensuite rejouer le message et son code le mardi sans que Bob ne puisse mettre son authenticité en doute.

Il existe principalement deux approches pour assurer la fraîcheur des messages. La première approche consiste à inclure un indice i dans chaque message en plus de son contenu. Ainsi, Bob enregistre

tous les messages envoyés par Alice et refuse un message dont l'indice est le même que celui d'un message qu'il a déjà reçu précédemment. Bob peut aussi détecter la perte d'un message (qui peut être accidentelle ou due à une action de Charlie) s'il reçoit les messages d'indices i et $i + 2$ mais pas le message $i + 1$. Le principal défaut de cette approche est qu'elle demande de garder les indices reçus ou du moins l'indice courant (si on utilise un compteur). La deuxième approche pour assurer la fraîcheur consiste à introduire une référence temporelle (*timestamp*) dans le message. Il faut pour cela que les deux interlocuteurs soient synchronisés. Le message sera ensuite refusé si le *timestamp* reçu n'est pas contenu dans un intervalle de temps défini par le protocole. Cependant, cette approche a deux défauts : elle requiert une synchronisation des horloges et elle permet des rejeux pendant la fenêtre d'acceptation.

5.8 Matériels didactiques et références bibliographiques

- [1] MIHIR BELLARE, RAN CANETTI AND HUGO KRAWCZYK, *Keying Hash Functions for Message Authentication*, CRYPTO 1996 : 1-15, 1996. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.8430&rep=rep1&type=pdf>
- [2] MIHIR BELLARE, JOE KIILIAN AND PHILLIP ROGAWAY, *The Security of the Cipher Block Chaining Message Authentication Code*, J. Comput. Syst. Sci. 61(3) : 362-399, 2000. <http://www.sciencedirect.com/science/article/pii/S002200009991694X>
- [3] CLAUDE SHANNON, *Communication Theory of Secrecy Systems*, Bell System Technical Journal 28 (4) : 656-715, 1949. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6769090>

Ce document a été rédigé par un consortium regroupant des enseignants-chercheurs, des professionnels du secteur de la cybersécurité ainsi que l'ANSSI.



L'ensemble des documents est distribué sous licence ouverte Etalab V1.